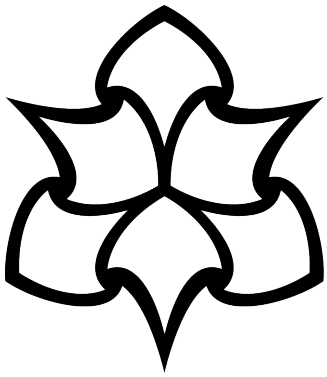

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received August 12, 2020, accepted August 14, 2020, date of publication August 19, 2020, date of current version September 1, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3017797

A Semantic and Syntactic Similarity Measure for Political Tweets

CLAIRE LITTLE[1], (Member, IEEE), DAVID MCLEAN[2], KEELEY CROCKETT[2], (Senior Member, IEEE), AND BRUCE EDMONDS[1]

[1]Centre for Policy Modelling, Department of Economics, Policy and International Business, Manchester Metropolitan University, Manchester M15 6BH, U.K.
[2]Department of Computing and Mathematics, Manchester Metropolitan University, Manchester M15 6BH, U.K.

Corresponding author: Claire Little (claire@clairelittle.com)

This work was supported in part by the European Union (EU) Horizon 2020 Program under Grant 822337.

ABSTRACT Measurement of the semantic and syntactic similarity of human utterances is essential in allowing machines to understand dialogue with users. However, human language is complex, and the semantic meaning of an utterance is usually dependent upon the context at a given time and learnt experience of the meaning of the words that are used. This is particularly challenging when automatically understanding the meaning of social media, such as tweets, which can contain non-standard language. Short Text Semantic Similarity measures can be adapted to measure the degree of similarity of a pair of tweets. This work presents a new Semantic and Syntactic Similarity Measure (TSSSM) for political tweets. The approach uses word embeddings to determine semantic similarity and extracts syntactic features to overcome the limitations of current measures which may miss identical sequences of words. A large dataset of tweets focusing on the political domain were collected, pre-processed and used to train the word embedding model, with various experiments performed to determine the optimal model and parameters. A selection of tweet pairs were evaluated by humans for semantic equivalence and correlated against the measure. The new measure can be used in a variety of applications, including for identifying and analyzing political narratives. Experiments on three diverse human-labelled test datasets demonstrate that the measure outperforms an existing measure, performs well on tweets from the political domain and may also generalize outside the political domain.

INDEX TERMS Semantic similarity, similarity measure, twitter, word embeddings.

I. INTRODUCTION

The ability to determine the similarity between two texts has applications in categorization, cluster analysis, dialogue systems, and document identification and matching. However, large-scale social media data present challenges when it comes to automating these processes. The ability to automatically identify content on a particular theme, or find similar (or dissimilar) text, has many applications and may be crucial to understanding and identifying the various narratives on social media platforms.

Twitter is a microblogging and social networking platform where users interact, and post messages known as tweets. Users may post their own tweets, ''like'' other users' tweets, retweet (or share) tweets, and quote or reply to tweets. Tweets are limited to 280 characters (increased from 140 in 2017), and this may include words, emojis or hashtags. Twitter reported an average of 330 million active users per month in the first quarter of 2019 [1], and the platform is often used for political discussion, playing a ''prominent role in how politicians, media outlets and advocacy organizations promote their agendas and engage with political issues'' [2].

Twitter allows users to share opinions and can allow politicians to easily speak directly to voters, about events as they are happening, bypassing the media that might otherwise filter or frame their content [3], [4]. Its use was credited with playing a part in the 2016 election of Donald Trump [5], whose frequent, and sometimes controversial, Tweets generated much free media coverage [6]. Twitter also plays a role in the spread of misinformation; this has become particularly evident concerning content around COVID-19 [7], [8] where misinformation about fake cures and treatments have contributed to accidental deaths [9].

The development of the Tweet similarity measure, which is the focus of this paper, grew out of a project to identify populist narratives within a political dataset. Populism has been

The associate editor coordinating the review of this manuscript and approving it for publication was Nilanjan Dey.

on the rise for a number of years [10], [11] and social media can allow populists a direct way to spread their message [3]. Agreement on the definition of populism has been notoriously difficult [12], however [13] describes it as a ''thin-centred ideology'' (in contrast to a fully formed ideology, such as socialism) whereby there is an antagonistic division between two homogeneous groups: ''the pure people'', who are necessarily good; and ''the corrupt elite''. A populist believes that politics should be an expression of the ''general will'' of the people and may therefore speak on behalf of ''the people'' and rail against ''the elite''.

Detecting a populist narrative from an automated machine perspective is a difficult challenge based upon the choice of language used and its semantic meaning in the context of the narrative. The less content that is available, such as in a tweet, the more difficult it is to automatically determine. In order to identify populist narratives a Word Embedding Model (WEM) [14] was trained on a dataset of tweets collected from the political domain and then employed to identify keywords that might be used within these narratives based upon a graph model of populist ideology [15].

In this paper we build upon this work with WEMs trained upon a political dataset and present a new Tweet Semantic and Syntactic Similarity Measure (TSSSM) which has been developed for measuring the similarity of political tweets. A similarity measure trained on political data may allow the identification and analysis of the various political themes (including populism) that exist over time and the ability to track changes in public opinion.

A large dataset of tweets from the political domain was collected, analyzed and pre-processed. The semantic element of the similarity measure utilized a WEM, and experiments to determine the optimal parameters for this were performed. A WEM was used in order to take advantage of the large amount of information contained in the dataset; the model could recognize entities, such as politicians, places and events. The syntactic element of the measure considers the presence of identical syntactic sequence of words, and syntactical features. Since tweets often contain quotes and repetition, a function to identify these is useful and complemented the semantic side of the measure. Utilizing a human-labelled training dataset the ideal parameters and weightings for the overall measure were determined; the final measure was evaluated using three human-labelled datasets.

The data collected as part of this study focusses on the political domain, containing tweets about Brexit, or ''British exit'' which refers to the United Kingdom (U.K.) leaving the European Union (E.U.). A referendum, which prompted much debate, was held in June 2016, and the U.K. voted to leave by a margin of 52% to 48%. However, there was much division and argument around this result, which meant that the process of leaving the E.U. became delayed. The U.K. formally left the E.U. on 31st January 2020, but with negotiations over future trade deals and a future relationship with the E.U. still to be completed. Brexit was chosen as the focus for data collection because it had been the pre-eminent theme in British politics for several years. Throughout this period there was, and continues to be, much traffic on Twitter discussing the various opinions and events surrounding the Brexit process. This meant that a collection of tweets focusing on Brexit provided a rich political dataset, with various themes and opinions, and changes over time.

A. RESEARCH QUESTIONS AND CONTRIBUTIONS

The overall research questions addressed in this work consider whether a similarity measure can be derived for measuring the similarity of political tweets that embodies both semantic and syntactic information, and whether the developed similarity measure can generalize to different domains.

The main contribution of this paper is the novel Tweet Semantic and Syntactic Similarity Measure (TSSSM) based on a Word Embedding Model for rating the similarity of tweets in the political domain. Although the measure and the WEM proposed in this paper were developed using data from the political domain, with the aim of identifying populist and political tweets and narratives, this research shows that the methods used can be generalized to other domains.

Using both semantic and syntactic elements in a similarity measure means that, as well as considering the similarity of meaning, the order of words, language usage and presence of specific features (such as hashtags or mentions) can also be considered. Whilst [16], [17] considered both semantic and syntactic elements of tweets, [16] did not consider the syntactical order of words, and [17] used topic modelling rather than a WEM and was designed for tweets in the Arabic language. TSSSM utilizes minimal pre-processing and can identify semantic relationships and common syntactical features and sequences. This is particularly important as tweets may often contain direct duplication (whole passages that are identical) which semantic measures alone may not detect [18].

The proposed Tweet Semantic and Syntactic Similarity Measure, TSSSM, has applications in various fields; for example, analyzing the spread of misinformation [8], [9] and ''fake news'' [19]. Most obviously it may be used as part of a clustering algorithm to determine similarity. It can also be used to validate clustering results, by determining whether tweets contained in the same cluster are actually similar. It may be particularly useful in social science research, allowing researchers the ability to select a tweet (or tweets) of interest and then identify other tweets that contain a similar message. This would be effective even where they do not necessarily share the same words and would be more sophisticated than simply searching on keywords or hashtags.

B. PAPER OVERVIEW

Section II describes related work in the holistic development of semantic similarity measures and their application in the realm of social media. Section III describes the collection of a political dataset from Twitter and the pre-processing methodology applied to the tweets. It also includes a description of the creation of a new human-labelled test dataset,

where 35 human raters evaluated the similarity of 32 tweet pairs within a range of similarities; a description of the other training and testing pairs utilized is also included. The development of word embedding models is described in Section IV and Section V outlines the development of the new TSSSM, together with an illustrative example. Section VI details the results on the testing datasets and compares the performance of TSSSM to a microblogging similarity measure known as TREASURE [16].

II. RELATED WORK

Whilst there are examples of short text similarity measures, there are few that focus specifically on tweets, nor that utilize a semantic and syntactic element, and even fewer that consider a political domain.

Semantic Text Similarity (STS) measures the degree of semantic equivalence between two texts; this may range from exact semantic equivalence to complete un-relatedness, with a range of nuanced shades of similarity in between [20]. Two texts might be semantically similar but share no common words. The yearly SemEval tasks have included some tasks with tweets [20], but whilst there is much work on sentiment analysis of tweets (for example, [21]–[24]), there is little work focusing on tweet similarity. However, [17] utilized paraphrase identification and topic modelling for semantic analysis to measure the similarity between Arabic news tweets, and TREASURE [16] considered semantic features (using a Word Embedding Model) and syntactic features (counting Parts of Speech tags and Twitter-specific features) to measure the similarity of political tweets. Reference [25] proposed classifying tweets based on a hybrid approach using sentiment analysis, fuzzy logic and semantic similarity using Wordnet.

There are a number of general short text similarity measures [18], [26], [27]. Latent Semantic Analysis (LSA) [28] is a statistical technique based on the analysis of word co-occurrence frequencies in large corpora and has been applied to various lengths of text [29], [30]. However, these measures do not necessarily extend well to tweets. Traditional semantic similarity measures were originally developed to determine the similarity of well-constructed short texts or sentences [31]. Tweets often have poor grammatical and syntactical structure, and there are major problems associated with the use of informal language (such as slang, textspeak, grammatical errors, and abbreviations), which can make them more difficult to analyze. This challenge is significant due to the application potential; whilst tweets do contain general sentences, they also contain additional features, such as hashtags, mentions, URLs and provenance information.

In terms of short text measures, [26] proposed a sentence similarity method using an edge-counting based technique between joint words from the two compared sentences after removing stop-words. Their method weighted the overall similarity by inclusion of word order calculation. STASIS [18] and [27], [32] proposed a knowledge-based approach that relies on graph traversal techniques applied to the Wordnet [33] taxonomy, which is composed of a hierarchy of noun synsets (synonyms). The knowledge embodied in the graph (e.g. path length, depth, and common subsumer) was used to compute sentence similarity through finding similar words in each of the sentence pairs under consideration and including other factors such as the Information Content (IC) from a corpus [18] and word order. IC is computed using a formula that considers the set of synsets in WordNet, and a constant that represents the total number of concepts in WordNet. However, such measures rely on searches through graph traversal and are computationally expensive. They are also too dependent upon well-formed English to be useful for tweet similarity. Although WordNet does include other word types, it is heavily biased towards nouns (the main ontology in WordNet is a hierarchy of noun synsets) and therefore the structures for other words are far less knowledge rich than for nouns.

Reference [31] performed an extensive review of short text similarity measures, selecting and comparing Bag of Words (BOW), LSA and STASIS in an experiment using the SemEval Tweet-News dataset [20] of 750 human annotated pairs, and found that while semantic-based measures performed better than keyword-based measures, further work would be required to develop measures that can handle noisy microblogging data.

Word Embedding models use artificial neural networks to learn a distributed representation of word co-occurrence information from a large corpus [34], and have shown significant improvements in the performance of many Natural Language Processing (NLP) applications such as sentiment analysis [21], [22], text classification [34]–[36] and recommendation [37]. This technique can be applied to a specific context by training on a large contextually appropriate and representative corpus. Word Embedding has been successfully applied in many domains including named-entity recognition [37], [38] and the political domain, where [39] trained a word embedding model on political tweets in order to utilize it in future semantic similarity measures and cluster analyses, and [40] used word embeddings to classify tweets collected during the Venezuela and Philippines general elections, finding that better performance was obtained where the background data aligned with the data to be classified.

III. FORMULATION OF A DATASET

A large dataset of 85,915,642 tweets pertaining to Brexit, referred to as the ''Brexit'' dataset, was collected via the Twitter streaming API. The Tweepy [41] and Pymongo [42] Python libraries were utilized, and the data stored in a nosql (mongodb) database. Data collection ran from 1st April to 18th December 2019 and was continuous, barring breaks of two weeks in June (1st to 15th) and August (10th to 29th), and any minor interruptions (such as API downtime or internet outages). Data collection began just after the initial deadline for Brexit (29th March) had been missed, and covered events in the aftermath, the second missed Brexit deadline (31st October) and the U.K. General Election (12th December).

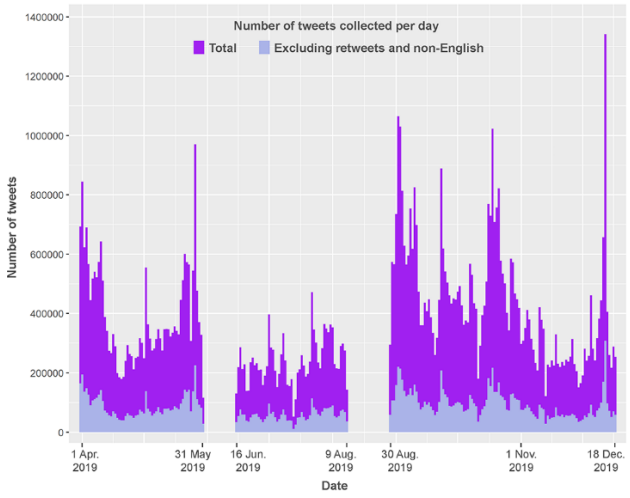

FIGURE 1. Bar plot of the number of tweets collected per day compared to the number where retweets and non-English tweets were excluded.

A filter was utilized to capture tweets containing one or more of the base keywords (''brexit'', ''nodeal'', ''peoplesvote'', ''backstop'', ''prorogue''). Keyword variations were also included (such as ''no-deal'', or ''prorogation''), and the filter could also capture the keywords in hashtag form. The filter returned tweet matches even where the keywords were not contained in the main body of text. This is because the keywords could be detected within the tweet metadata (not just the main text) and this meant that a more nuanced dataset was collected; it captured tweets where people were talking about Brexit (i.e. they were replying to or quoting something on that subject) but did not necessarily mention any of the keywords in their own text. The keywords pertaining to ''backstop'' and ''prorogue'' were added from 30th August onwards to reflect the changing political context surrounding Brexit at that point.

In order to prepare the data for analysis all retweets and tweets not in English were removed. The dataset contained tweets written in 63 different languages (as labelled by Twitter), 94% (80,737,805) of which were in the English language. However, the focus was on English language tweets, therefore those in other languages were not required. Retweets made up just under three quarters (74.6%, or 64,124,243) of all tweets, however, since retweets are direct duplicates, they were removed to avoid redundancy of data. The resulting dataset contained 19,217,186 tweets, that is 22.4% of all (85,915,642) tweets collected.

Fig. 1 shows the frequency of tweets collected per day compared with the frequency when retweets and those not in English were excluded. Fig.1 highlights there was great variation in the frequency of tweets collected per day; this was perhaps reflective of the ever-changing political climate around Brexit, as the peaks generally appeared to coincide with political events happening at the time. However, the number of tweets available on the streaming API are limited and Twitter provide no definitive information on how large a sample of tweets are available, nor how they are sampled; it is thought that at most 1% of the overall tweets on Twitter can be collected at any given time [43]. The collected data can therefore only be taken at face-value and it is supposed that patterns noticed in tweet frequency are due to volume rather than limitations, or a quirk, in the API. The maximum number of tweets collected on one day was 1,341,470, on 13th December; this was the day after the U.K. general election, when the results were revealed.

The 19,217,186 tweets were tweeted by 1,690,955 unique users; meaning each user tweeted on average 11 times. However, analysis suggests a smaller group of users appear to be responsible for a large number of tweets; just under half of all users (820,469 or 48.5%) tweeted only once, whereas 1,173 (0.07%) unique users tweeted at least 1,000 times each.

Just over half (51.2% or 9,834,836) of tweets mentioned other users (using the ''@username'' convention). The most mentioned user was Boris Johnson (610,433 mentions), followed by the Brexit Party (319,625 mentions) and Jeremy Corbyn (304,672 mentions). Just under a quarter of tweets (23.4% or 4,495,816) used hashtags. #brexit appeared most frequently (2,519,333 times), followed by #peoplesvote (350,547 uses) and #eu (154,426 uses).

A. TWEET PRE-PROCESSING METHODOLOGY

For use in the Word Embedding Model (WEM), the tweets were pre-processed using the steps illustrated in Fig. 2. Pre-processing is necessary in order to standardize the text and remove noise. For example, the tweet ''the weather is . . .AWFUL. . .today!!!:(:(:('' contains capitalization, extra spaces, punctuation and symbols; it would be pre-processed to ''the weather is awful today'' which still retains the meaning. Pre-processing tweets for use in a WEM is an important step since tweets tend to contain noise and non-standard language, which if retained may increase the vocabulary size and computational cost of the resulting model [37].

As illustrated in Fig. 2, duplicates were removed at the beginning, as well as at the end of the process, to avoid unnecessary processing. Variations of ''rt:@username'' and ''via: @username'' were removed, as these tended to appear in retweets (but where Twitter had not identified them as such); if the remaining text was a duplicate then it would be removed in the final duplicate removal step.

The pre-processing was designed to retain as much of the information and structure contained within the text as possible. It aimed to remove punctuation and symbols without unintentionally altering the meaning. For example, if all commas were replaced with no space, this would be optimal for numbers (''123,456'' becomes ''123456'' and the meaning is retained) but not necessarily for text. It was observed that Twitter users often used commas with no space, meaning that in some cases words would be joined together if commas were removed (for example, ''England,Scotland, Wales'' would become ''EnglandScotlandWales''). Therefore, different rules were applied depending upon the usage; commas between letters were replaced with a space, commas

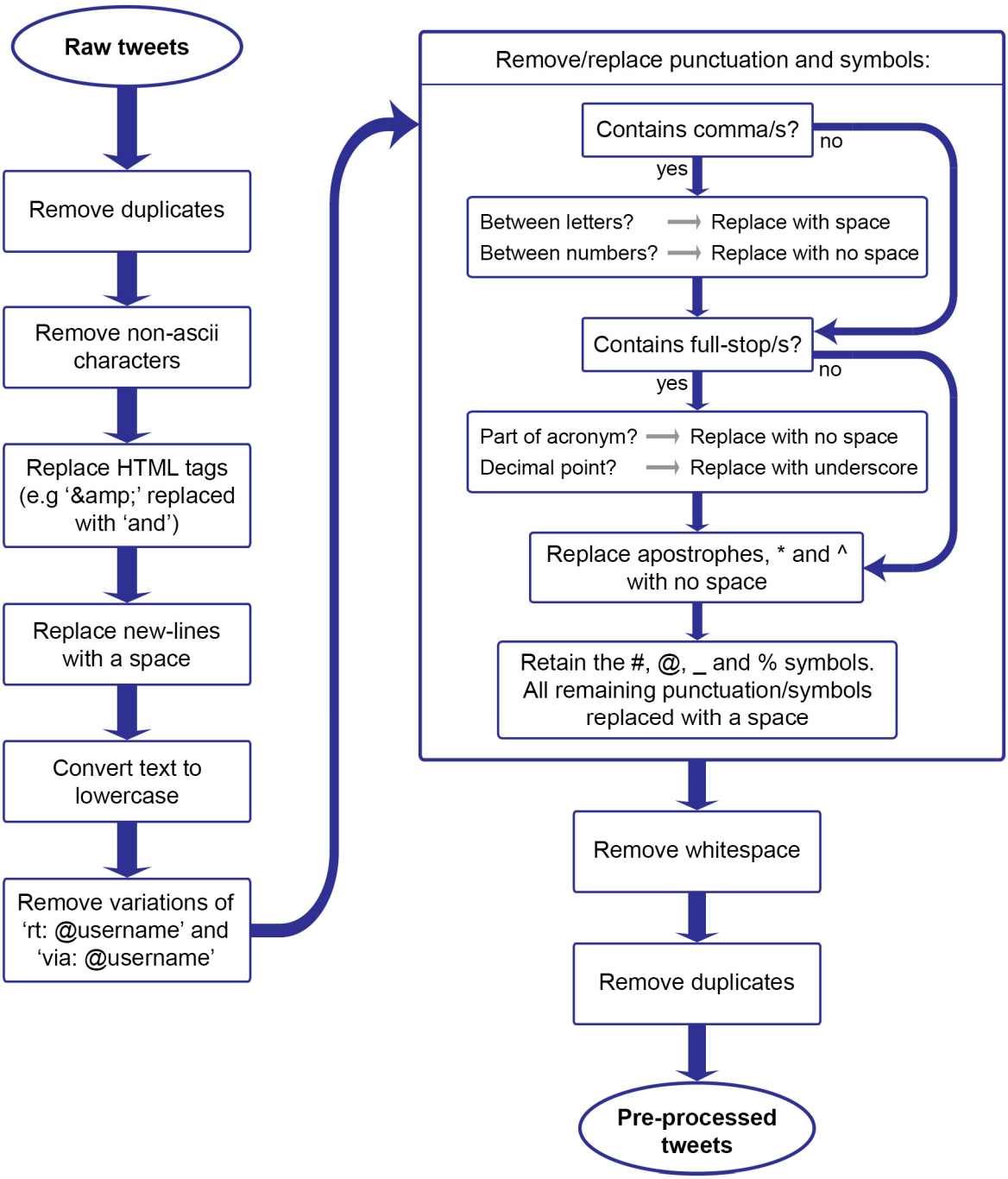

FIGURE 2. Flow chart illustrating the steps performed as part of the tweet pre-processing method.

between numbers with no space. This consideration was also given to other punctuation.

The "#", "@", "_" and "%" symbols were retained, as hashtags, usernames (which can contain underscores) and the "%" symbol (percentages were much quoted in the Brexit debate) were deemed important to the analysis. Twitter users may refer to people (or things) in many different ways, using hashtags, usernames or plain text. For instance, "Boris Johnson" could also be referred to as "@borisjohnson", "#boris" or other variations. A WEM can automatically learn relationships between words and identify words that are used in a similar context, meaning that those variations can exist within the data and the model will identify them as similar. A WEM may also be utilized for identifying common spelling mistakes; the ability to identify spelling mistakes, or variations of words, is particularly useful for tweet data, where users may employ non-standard language. Tweets with less than five words were removed as it was felt that it would be

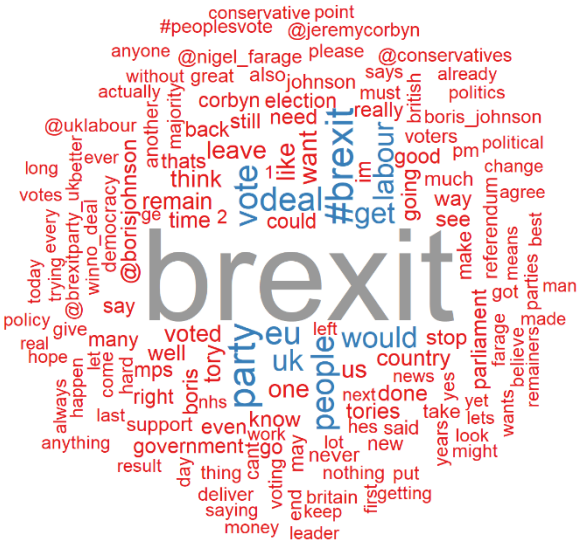

FIGURE 3. Word-cloud visualization of the most frequently used words in the corpus, excluding stop-words.

difficult to convey much useful information in such a short text, and the ''window'' of words for training the WEM would also be small. This resulted in the removal of 520,430 tweets.

After pre-processing, and the removal of short tweets, 16,549,251 (86.1% of 19,217,186) unique tweets remained. The Gensim [44] Python module was then utilized to detect bigrams; this detects commonly occurring phrases and links them together into one token. For example, a phrase such as ''general election'' becomes ''general_election''. It is particularly useful for detecting names (e.g. ''boris johnson'' becomes ''boris_johnson''). The default settings for bigram detection were utilized (min_count = 5, and threshold = 10). The 'min_count' parameter ignores all words with a total count below the value (i.e. 5); the 'threshold' parameter represents a score for forming the phrases, where higher means fewer phrases. Following bigram detection, the mean length of tweet, in terms of number of words/tokens per tweet, was 26.5 words (prior to this it was 27.8 words).

The entire Brexit corpus consisted of 460,266,017 words. Including bigrams, there were 2,565,087 unique words in the corpus. The high number of unique words was likely due to the inclusion of usernames. Just over half (1,306,067, or 50.9%) of the words in the corpus were used only once. The five most frequently used words (together with the number of uses) were: ''the'' (20,786,374); ''to'' (12,170,935); ''brexit'' (11,430,592); ''a'' (9,853,788): and ''and'' (9,511,685).

Aside from ''brexit'' the most frequently used words tended to be stop-words, as might be expected. For some NLP methods stop-words may be removed from a corpus, however they were retained for this work as it was deemed important to maintain the structure of the text for building the WEM. However, for informational detail the five most frequently used words, excluding stop-words (as defined by the NLTK Python package [45]) were, with the number of uses in parentheses: ''brexit'' (11,430,592); ''#brexit'' (2,302,694); ''party'' (2,276,116); ''deal'' (1,787,607); and ''vote'' (1,744,770).

Fig. 3 contains a word-cloud visualization of the most frequently used words in the corpus with stop-words removed. The word-size is scaled by its appearance frequency in the corpus. It illustrates that the word ''Brexit'' was by far the most frequently used, as might be expected given its use as a keyword in the data collection process. It also provides a view of the other commonly used words within the corpus.

B. TRAINING AND TESTING DATASETS

In order to develop the similarity measure, labelled training and testing data was required. The training data allowed the optimal WEM and measure parameters to be chosen. Once the optimal parameters were determined TSSSM was evaluated using the testing data which was not used in the training process and therefore able to provide an unbiased view.

The evaluation of a semantic similarity measure requires benchmark datasets derived from similarity ratings provided by humans [30]. That is, humans are asked to rate pairs of texts for semantic similarity (generally, the average of their ratings is taken) and the output from the measure is compared to these human ratings. A well-performing measure will achieve scores, or ratings, close to the human ratings. Whilst there are human-labelled datasets that include short text example pairs, there are few that include tweets. However, three sets were utilized: a set of 32 pairs developed for this project using the Brexit corpus; a set of 30 pairs, ''EU-referendum'' taken from [16]; and a larger set of 750 pairs from the SemEval 2014 [20] ''Tweet-News'' dataset. The majority of these pairs were reserved for testing, and a smaller set selected for training.

The Tweet-News pairs, selected from a larger set [46], consist of a news headline paired with a Twitter comment on the particular headline, with human ratings provided by the Amazon Mechanical Turk crowdsourcing service. The Tweet-News dataset therefore does not contain two tweets in a pair, but the pairs were considered to be close to resembling two tweets, and hence useful for training the similarity measure and testing its generalizability. However, it was noted that whilst some of the tweets contained hashtags, few contained mentions (and the paired headlines contained neither). Therefore, the Tweet-News pairs could not be used solely for training the measure as examples typical of paired tweets would still be required.

1) SIMILARITY SCALE

Each of the training and testing datasets utilized the same similarity scoring scale [20], using ratings on a scale of 0.0 to 5.0, and detailed as such:

- 5.0 The two tweets are completely equivalent as they mean the same thing
- 4.0 The two tweets are mostly equivalent, but some important details differ
- 3.0 The two tweets are roughly equivalent, but some important information differs/missing

2.0 The two tweets are not equivalent, but share some details
1.0 The two tweets are not equivalent, but are on the same topic
0.0 The two tweets are on different topics

With this scale human raters could use finer degrees of similarity if they preferred, for example, a value of 3.5 or 1.8. Using a ratio scale to measure semantic similarity allows an absolute zero point on the scale and the setting of an upper bound, which is common in word similarity measures [30]. A scale of 0.0 to 5.0 was chosen as it allowed human raters to clearly indicate no (0.0) or maximum (5.0) similarity whilst also allowing fine-level evaluation of the similarity between these bounds. This scale was also used in the yearly SemEval Semantic Textual Similarity tasks and was used for the Tweet-News [20] and EU-Referendum [16] datasets used for training and testing TSSSM. Adopting the same scale for the Brexit dataset ensured consistency of comparison.

2) TRAINING DATA

A random sample of 63 pairs from the Tweet-News set combined with 21 from the Brexit set, were utilized for training TSSSM. The 21 Brexit pairs were candidate pairs for the Brexit test pairs set (details in the following section, III.B.3) but were not selected. Their inclusion in the training data was to ensure that the training set contained examples of typical tweet pairs taken directly from the corpus, and where both in the pair were tweets (Tweet-News contains only one tweet in each pair, and they are out of corpus). The ratio of 1 Brexit pair to 3 Tweet-News pairs was chosen to allow a large enough sample of labelled pairs to train on, whilst allowing representation of in-domain tweet pairs.

3) TESTING DATA

In order to test directly on the Brexit corpus, a human-labelled benchmark dataset was developed from tweet pairs contained in the corpus. This resulted in 32 human-rated pairs of tweets, referred to as the ''Brexit'' pairs.

The Brexit pairs were prepared by: manually selecting an initial 55 pairs of tweets with varying levels of similarity; convening a panel of experts (utilizing the methodology of [30]) to rate the similarity as high, medium or low; and selecting only those pairs where the human-raters agreed (either unanimously or by majority) on the level of similarity. Two calibration pairs were included (one would be expected to have full similarity and the other no similarity). The dataset therefore consisted of 2 calibration pairs and 30 other pairs that aimed to have varying levels of similarity.

The initial 55 candidate pairs were manually selected by choosing 1-hour timeframes where there were peaks in the number of tweets collected. The themes on Twitter are often time-specific with users tweeting or replying to events as they are happening; hence where there was a peak, many of the users were responding to the same event, which aided in determining tweet pairs. For the months of April, May, July, September and October (June/August were excluded as data collection did not cover the entire month) the hour where the most tweets had been collected was determined. These each had one or two clear themes, for instance, the hour of 11pm to 11:59pm on 26th May contained much discussion of the European Election results, which had been announced at around 10pm that day.

For each of the five selected 1-hour timeframes, candidate pairs, with varying levels of similarity, were selected based around the themes. Some pairs were considered very similar in meaning, whereas others might contain a common name or words (but with different levels of similarity), whilst others were deemed to have little or no similarity. Whilst similarity can be subjective, the use of the 'expert panel' to determine the final 32 pairs aimed to ensure that there was some agreement as to the level of similarity.

21 of the candidate pairs that were not selected for the final test set were utilized as extra training pairs; this was in order to provide examples of tweet pairs directly from the Brexit corpus during the training process. As noted previously the Tweet-News training pairs were not all representative of typical tweets (lacking hashtags, mentions, etc.). Each of the 21 extra Brexit pairs had similarity ratings via the expert panel and no unanimous disagreement on the similarity level.

The final 32 tweet pairs for the test set were randomly ordered and anonymously rated via an online survey. This had 35 respondents; all were aged 18 or over and had some interest in politics. To try and ensure consistency of scoring, each tweet pair was displayed on a separate page and the similarity scale was displayed below as a reminder.

Fig. 4 plots the individual scores that each of the pairs received. They are ordered by the mean similarity score of each (high to low), and with a line displaying the mean score. Aside from the two calibration pairs which had mean scores of 0.0 and 4.8, there was variation in how the respondents scored the pairs. The zero-similarity pair (pair 32) performed as expected, with all respondents assigning a score of zero. However, the full similarity pair (pair 1), which contained identical texts differing only in the addition of ''#sky #news'' at the end of one of the pairs, did not receive unanimous scoring. Whilst 63% (22 out of 35) of respondents gave a full similarity score of 5.0; almost a third felt that the presence of the hashtags reduced the similarity. However, 89% (31 out of 35 respondents) gave a score of 4.8 or above. The four lower scores may have been given in error, or it may be that the respondents felt that the presence of hashtags in one of the pairs changed the meaning.

Fig. 4 highlights the difficulty in assessing the semantic similarity of two texts. Whilst some of the pairs were generally rated consistently by the majority of respondents (for example, pair 6 had no scores below 2 and was consistently around 3 or 4, and pair 31 had no score above 2.5 and was consistently rated low), others (such as pair 15) received scores that spanned the whole range.

Despite the variation in scoring it was deemed that all respondents' views should be included. The candidate pairs indicated that, overall, the survey worked as it should.

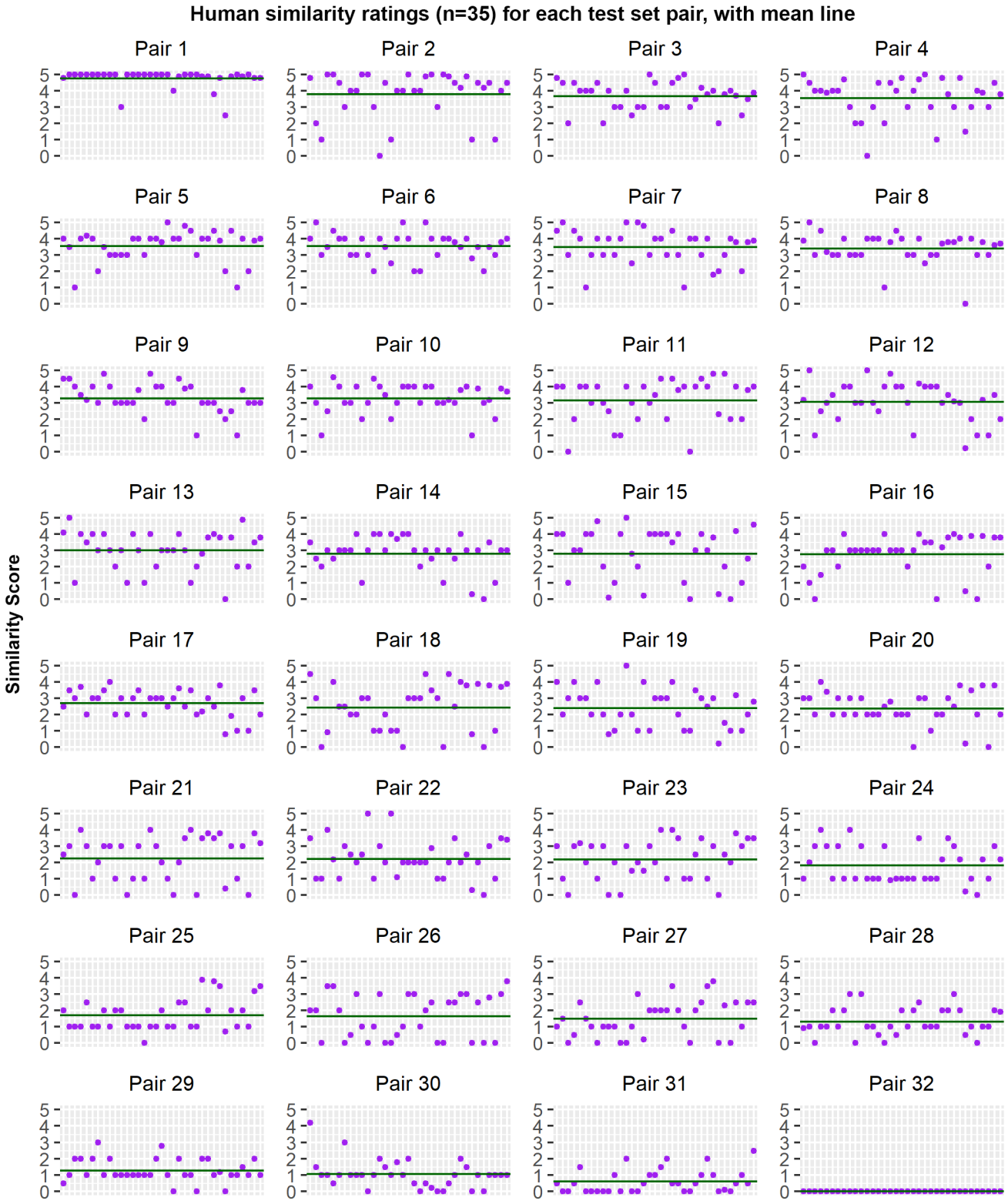

FIGURE 4. Plots of the individual human similarity scores (n=35) assigned for each of the 32 Brexit test set pairs, ordered by mean similarity (high to low) with a line displaying the mean. This indicates the divergence of human opinion when judging tweet similarity.

Therefore, the overall similarity score for each of the 32 Brexit pairs was calculated as the mean of the respondent's judgements.

Due to data protection and ethical policy the text for the tweet pairs is not published. However, the tweet IDs and scores will be made available upon request.

Together with the Brexit set, two further test sets were utilized in order to evaluate TSSSM. The ''EU-Referendum'' set [16] contained 30 human-rated tweet pairs collated from a dataset collected around the Brexit referendum vote in 2016. Its tweets were therefore on the same political domain but collected three years earlier and allow evaluation using pairs each containing two tweets that are in a similar domain but not contained in the corpus. The remaining Tweet-News [20] pairs (n=687) that were not used for training were also utilized, in order to determine how TSSSM generalizes on pairs outside of its domain.

IV. DEVELOPMENT OF WORD EMBEDDING MODELS

TSSSM utilizes a Word Embedding Model (WEM) [14] in order to determine the semantic similarity between tweet pairs. Word Embedding is a natural language modelling technique that is designed to map words or phrases from a vocabulary on to a corresponding vector of numerical values. It works on the premise that words that regularly occur together in the text will also be in close proximity in the vector space. A WEM automatically learns the semantic relationships between words.

A WEM utilises a shallow neural network and assigns each word in the corpus to an n-dimensional vector (n is user-defined). The vectors can then be used to demonstrate linear relationships between words, which may sometimes have intuitive results, such as in the example of [14], where it was shown that vector(''King'') – vector(''Man'') + vector(''Woman'') resulted in a vector closest to the word representation of ''Queen''. The vectors also allow the calculation of how close two words are in the vector space, or how ''similar'' they are based on context, using cosine similarity.

The WEM is utilized in TSSSM in order to directly determine the semantic similarity between two words. The value returned, between 0 and 1, scores how frequently two words are used in the same context. For instance, ''brexit'' and ''#brexit'' would be expected to have a high score, closer to 1. Whereas ''brexit'' and ''potato'' would be expected to have a much lower score. However, a consideration when using WEMs is that words which have seemingly opposite meanings may have a high score (for instance, ''agree'' and ''disagree''). This is because if they are frequently used in similar contexts, their vectors may be similar. This means that a WEM may not be ideally suited for use as a thesaurus (for example) but allows a level of subtlety for a similarity measure, in terms of considering the semantic relationships between words.

As there is little definitive literature on the optimal parameters for a WEM (and, in particular, its use in a similarity measure), eight different models were built utilising varying parameters, with the optimal model for the measure determined during the training process. The WEMs were built with the Python programming language using the Gensim [44] implementation of the Word2Vec algorithm [47].

For each model the parameters as utilized in [47] were followed – that is continuous Skip-gram architecture using hierarchical softmax, negative sampling, a word vector of size n=300, and down-sampling of frequent words. The two parameters that were varied between models were window size and the minimum word count threshold, as it was likely that these parameters would have most effect upon the model's effectiveness in a similarity measure given that they directly affect the semantic relationships and the size of vocabulary. The window size controls how many surrounding 'context' words are considered – a window size of 5 would consider the five words before and five words after the target word. Window size can affect performance in various tasks [38], [40], [48]; with findings that a smaller window size is optimal in named entity recognition [38], and dependency parsing [48], but a larger window preferred for classification [40]. Models with window sizes of 3 and 5 were built.

TABLE 1. Parameters for the word embedding models.

Parameter	Value
Architecture	Skip-gram
Number of dimensions	300
Negative sampling	5
Down-sampling of frequent words	0.001
Training epochs	5
Window size values	3, 5
Minimum word count values	2, 3, 5, 10

The minimum word count parameter allows the model to discard words that appear rarely in the corpus; this may speed up model creation by allowing a smaller vocabulary. For instance, words that appear only once in a large corpus, are likely to be spelling mistakes (or perhaps, in this case, usernames); setting the minimum word count parameter to 2 would mean that these are excluded. Varying this parameter allowed a methodical consideration of the effect of vocabulary size on the performance of TSSSM. Therefore, models were built with the minimum word count threshold set at 2, 3, 5 and 10. Varying the parameters produced eight models. That is, four models with the window size set to 3, and a minimum word count of 2, 3, 5, or 10, and four models with the window size set to 5, and a minimum word count of 2, 3, 5 or 10. The range of parameters tried for the models are in Table 1.

Model creation consisted of two steps: building the vocabulary table, and training. The process of building the vocabulary table incorporates the minimum word count parameter and involves analyzing the entire corpus, filtering out rare words and down-sampling common words, in preparation for training. The training utilized 5 epochs in order to enhance the quality of the models. Table 2 lists the number of unique words in the corpus for each individual model (labelled from A to H). The time to train for each model ranged from 72 to 104 minutes, with a larger window size taking longer (as more words need to be considered around the target word) and a smaller corpus training quicker. It should be noted that for use in TSSSM, the model is trained once offline and once trained it is computationally efficient to access. Table 2 indicates the number of unique words in the corpus for each of the models trained.

TABLE 2. Number of unique words in the corpus for each WEM.

Model	Number of unique words in corpus
A: window size = 3, min. word count = 2	1,259,020
B: window size = 3, min. word count = 3	958,359
C: window size = 3, min. word count = 5	723,761
D: window size = 3, min. word count = 10	429,326
E: window size = 5, min. word count = 2	1,259,020
F: window size = 5, min. word count = 3	958,359
G: window size = 5, min. word count = 5	723,761
H: window size = 5, min. word count = 10	429,326

To illustrate the semantic relationships that a WEM can describe (using Model D as an example) the vector(''@conservatives'') – vector(''@theresamay'') + vector(''@jeremycorbyn'') resulted in a vector closest to the word representation of ''@uklabour''. This means that the model identified that Jeremy Corbyn is to the Labour Party as Theresa May is to the Conservative Party (at the time of data collection Jeremy Corbyn was UK Labour Party leader, and Theresa May the Conservative Party leader). The model also found more general relationships, for instance, that Berlin is to Germany as Paris is to France.

The eight WEMs (A-H) were utilized within the training process of TSSSM to determine which was optimal for use in the measure.

V. DEVELOPMENT OF A TWEET SIMILARITY MEASURE

TSSSM is composed of two elements: semantic and syntactic. The semantic element utilizes a word embedding model to determine the level of semantic similarity between the words in the two tweets, and the syntactic element considers grammatical structure and the presence of syntactical features. Including the syntactic features means that word order is considered. Two texts could contain exactly the same words but have completely different meaning (for example, switching the subject and object within a verb clause) and in this case a semantic element alone would score them as identical when they are not; it is therefore important to consider word order [18]. The two semantic and syntactic elements were developed separately and then combined in the final training step.

A. SEMANTIC ELEMENT

The development of the semantic element of TSSSM required a WEM (of which 8 were tested), corpus weights (calculated from the word frequency in the corpus) and training data (the combined Tweet-News sample and Brexit training dataset). The training data was used to determine the optimal parameters. That is, the optimal WEM and similarity threshold (α, section V.A.2) for use in the WEM, whether stop-words should be included in the semantic match, and whether the words should be weighted using corpus weights.

The semantic comparison follows the methodology of [18] whose work focused on short text (but not tweets) and utilized Wordnet rather than a WEM. Reference [16] extended this method and used a WEM to create a measure known as TREASURE. TSSSM, proposed in this paper, builds upon the foundation of these works by creating a different algorithm, adding different features and parameters, and using a much larger corpus base for the WEM.

1) CORPUS WEIGHTS

Whether to include corpus weights was considered during development of TSSSM. The weights come from the Information Content of the corpus. With corpus weights included, those words occurring most frequently in the corpus were weighted down (as frequently occurring words may be less likely to contribute significantly to the tweet meaning), whereas more unusual words that appeared less frequently in the corpus were given a higher weight (as they may be more likely to contribute to the meaning). The weights were calculated as in (1), with $W(w)$ being the weight of word w, n the number of occurrences of the word in the corpus, and N the total number of words in the corpus (1 is added to avoid error with the logarithm).

$$W(w) = 1 - \frac{\log(n+1)}{\log(N+1)} \tag{1}$$

Using (1) all words in the corpus were assigned a weighting such that $W \in (0, 1]$. Commonly occurring words have a weight closer to zero, and rarer words a weight closer to 1. When the similarity measure encounters a word that does not appear in the corpus, and therefore might be considered unusual, it is assigned a weight of 1 (as in this case, the numerator, log(n + 1) would equal log(1) which is zero).

2) SIMILARITY THRESHOLD

TSSSM utilizes the WEM to determine how similar pairs of words from each tweet are. The WEM assigns a score between 0 and 1, with a value of 1 indicating two identical words and lower values indicating less similarity. In general, two words with seemingly little similarity will not receive a score of zero; they tend to be assigned a (low) score greater than zero. Therefore, a similarity threshold,

$$\alpha \in [0, 1] \tag{2}$$

was utilized, whereby word pairings with a similarity value below this threshold would be ignored. Discounting the similarity score of highly dissimilar words avoids introducing unnecessary noise to the semantic vector [16], [18]. A threshold of 0.3 was utilized by [16]. This meant that only pairs of words with a similarity score greater or equal to 0.3 were counted; scores below this would be set to zero.

The optimal similarity threshold, α, for TSSSM was determined during training, where various experiments were performed, including determining whether it would be advantageous to set a different (higher) threshold solely for stop-words. Values for α ranging from 0.1 to 0.7 were tested: a value too low would result in noise (allowing matches that were dissimilar) whereas a value too high may result in excluding similar matches.

3) METHOD

Each tweet is represented by the words, or tokens, it contains (as well as words and n-grams, these might also be numbers, or hashtags, etc.); for simplicity they will be referred to as ''words''. The words in each tweet pair are compared to each other, and the WEM is used to determine the level of semantic similarity between them. The semantic element of the measure does not take into account word order, this is considered by the syntactic element.

Tweets pairs were pre-processed using the steps detailed in section III.A, with the same bigrams applied. For each pair of pre-processed tweets, a joint word-set was created, consisting of all unique words from both tweets. That is, given two tweets, T_1 and T_2, their union, T is:

$$T = T_1 \cup T_2 = \{g_1, \quad g_2, \ldots, g_m\} \tag{3}$$

where g are the words and m the number of distinct words in the joint word-set. Excluded from the joint word-set was a small set of 15 frequently occurring words (an, and, as, at, be, by, cc, is, in, it, on, of, the, to, via) and single letters (a, b, c, etc. excluding ''I''). Because these words occur so frequently, and in so many contexts, they do not return useful semantic matches and so could be excluded. Similarly, single letters tended to represent noise and they also do not return useful semantic results. Experiments were performed during training to justify this.

As an example, the two texts ''we need another vote'' and ''the people need to vote again'' would have a joint word-set of $T = \{$we, need, another, vote, people, again$\}$.

In common with [18], lemmatization was not applied to the joint word-set as the WEM can effectively match similar forms of words. Reference [16] suggested this may result in sparse vectors, but it was not found to contribute to this in experiments.

Following the methodology of [16], [18] each individual tweet, T_1 and T_2, is compared to the joint word-set, T, and two vectors, V_1 and V_2, are created containing the semantic similarity between each tweet and the joint word-set. Words contained in both the tweet and the joint word-set are assigned a similarity score of 1, as they are a direct match. TSSSM will also match two words that differ only by the presence of a hashtag symbol, for example, #cat and cat, would also be assigned a score of 1. Each word that is in the joint word-set but not the tweet is then compared for similarity with all words in the tweet, using the WEM. The match with the highest similarity score is selected if it is above the threshold, α (determined during the training process). If none of the matches are above the threshold, α, then a value of zero is applied. If none of the words are in the WEM vocabulary, then they return a zero. This results in two vectors of semantic similarity, V_1 and V_2, representing each tweet.

During development of TSSSM, experiments were performed with and without corpus weights (i.e. the Information Content of the words) to determine whether their inclusion was optimal. Where the weights were included, a vector, W_T containing the corpus weights for each of the words in T was obtained; words not in the corpus were assigned a weighting of 1 (to reflect they were rare). The corpus weights were then applied by multiplying each of the semantic similarity tweet vectors V_1 and V_2, by the weight vector W_T, such that

$$s_i = V_i.W_T \tag{4}$$

where s_i is the weighted semantic vector. For experiments where the corpus weights were not utilized, this step was not included.

The overall semantic similarity between the two tweets is calculated as the cosine coefficient between the two weighted/unweighted semantic vectors:

$$S_{sem}(T_1, T_2) = \frac{s_1.s_2}{||s_2|.||s_2||} \tag{5}$$

For the unweighted version s_1 and s_2 are replaced with V_1 and V_2. The semantic similarity, S_{sem}, may take a value between 0 and 1, with a zero indicating no semantic similarity and a value of 1 indicating the two tweets are semantically identical.

4) TRAINING THE OPTIMAL MODEL

The training encompassed four parts:

- Determining the optimal WEM and similarity threshold (α)
- Determining whether excluding stop-words would yield better results
- Determining whether including stop-words but setting a higher similarity threshold (α) solely for stop-words would be optimal
- Determining whether the inclusion of corpus weights was optimal

In order to determine the optimal WEM and similarity threshold α, each of the 8 WEMs (A – H) were tested with thresholds ranging from $\alpha = 0.1$ to $\alpha = 0.7$. The semantic score, S_{sem}, was therefore calculated 56 times for each scenario.

In order to consider whether excluding, or requiring a higher level of similarity, for stop-words would be optimal, the same process was applied (all WEMs, A – H, and thresholds $\alpha = 0.1$ to $\alpha = 0.7$ were considered). Even though frequently used words are weighted down by the corpus weights (if used), it was considered that stop-words might still have undue influence on TSSSM. Therefore, three scenarios were considered: excluding stop-words altogether from the joint word-set (meaning they would not be checked for matches at all); excluding stop-words from the semantic matching process (but they were included in the joint word-set so a direct match would still count); or setting a higher similarity threshold just for stop-words. Where stop-words were assigned a higher similarity threshold experiments were performed with three different thresholds: $\alpha + 0.1$, $\alpha + 0.2$ or $\alpha + 0.3$. That is, the similarity thresholds for stop-words was 0.1, 0.2 or 0.3 above the threshold, α, used for regular words. Since there is no definitive list of stop-words, the stop-words as defined by NLTK [45] (n=179), spaCy [49] (n=326)

TABLE 3. Selected results for training including corpus weights, with optimal results in bold.

WEM Parameters	Threshold α	Including stop-words		Excluding stop-words		Threshold of α+0.1 for stop-words		Threshold of α+0.2 for stop-words	
		MSE	Cor.	MSE	Cor.	MSE	Cor.	MSE	Cor.
Model B: window = 3, min word = 3	0.1	2.055	0.698	1.943	0.703	2.054	0.698	2.033	0.699
	0.2	1.981	0.691	1.813	0.706	1.927	0.694	1.875	0.706
	0.3	1.366	0.665	1.231	0.686	1.251	0.691	1.183	0.707
	0.4	**0.937**	0.731	**0.962**	0.733	**0.868**	0.755	**0.853**	0.761
	0.5	1.242	0.751	1.383	0.734	1.270	0.756	1.276	0.757
	0.6	1.951	0.704	2.010	0.685	1.957	0.706	1.957	0.706
	0.7	2.246	0.683	2.322	0.658	2.246	0.683	2.246	0.683
Model C: window = 3, min word = 5	0.1	1.824	0.703	1.704	0.704	1.818	0.704	1.788	0.703
	0.2	1.709	0.689	1.562	0.693	1.648	0.687	1.595	0.698
	0.3	1.103	0.707	1.003	0.726	1.007	0.728	0.948	0.744
	0.4	0.975	0.729	1.025	0.730	0.905	0.757	0.908	0.758
	0.5	1.314	0.744	1.448	0.721	1.338	0.748	1.348	0.747
	0.6	1.911	0.712	1.970	0.691	1.922	0.711	1.922	0.711
	0.7	2.296	0.677	2.373	0.653	2.296	0.677	2.296	0.677
Model D: window = 3, min word = 10	0.1	1.625	0.696	1.521	0.692	1.607	0.696	1.570	0.699
	0.2	1.392	0.710	1.284	0.715	1.325	0.718	1.286	0.725
	0.3	0.996	0.716	0.991	0.715	0.963	0.723	0.907	0.740
	0.4	1.015	0.737	1.080	0.737	0.950	0.767	0.961	0.767
	0.5	1.431	**0.753**	1.508	0.739	1.451	0.759	1.453	0.758
	0.6	1.966	0.715	2.017	0.694	1.968	0.715	1.968	0.715
	0.7	2.296	0.677	2.372	0.653	2.296	0.677	2.296	0.677
Model H: window = 5, min word = 10	0.1	1.622	0.705	1.473	0.708	1.603	0.707	1.582	0.706
	0.2	1.405	0.704	1.261	0.710	1.364	0.704	1.303	0.714
	0.3	1.024	0.710	0.962	0.726	0.963	0.727	0.915	0.740
	0.4	0.977	0.742	1.035	**0.749**	0.916	**0.770**	0.905	**0.776**
	0.5	1.337	0.742	1.471	0.721	1.353	0.749	1.366	0.748
	0.6	1.898	0.722	1.959	0.700	1.911	0.721	1.911	0.721
	0.7	2.296	0.677	2.373	0.653	2.296	0.677	2.296	0.677

and Gensim [44] (n=337) were utilized. These experiments resulted in 1400 runs. The experiments were then repeated to exclude corpus weights, meaning that there were 2800 results in total.

The combined Brexit and Tweet-News training dataset (n=64) was utilized as labelled training data. Since the semantic element outputs a score (S_{sem}) between 0 and 1, and the training data has a score between 0 and 5, the semantic output (S_{sem}) was multiplied by five. The score was then compared to the labelled score to determine correlation and error rates. In general, correlation is considered the standard method of evaluating semantic similarity measures [30], However, as suggested in [31], calculating the error between the actual and estimated values is reasonable. Indeed, error rates are generally preferred to correlation for most machine learning applications. Given this, the Pearson correlation (cor.) and mean squared error (MSE) were calculated to provide a full picture of results. MSE is chosen to provide an insight into the degree of error. The optimal measure parameters would maximize the correlation and minimize the MSE.

A small selection of results is included in Table 3 and Table 4; they contrast the same four scenarios with the use of corpus weights (Table 3) and without corpus weights (Table 4). The stop-words are those as defined by Gensim [44] and for ease of reading the results for 4 out of 8 WEMs are displayed. For each experiment the MSE and correlation are displayed to 3 decimal places. The four scenarios displayed in both tables are the experiments where stop-words were included, stop-words were excluded before the joint word-set was formed (so they were not checked for a semantic match by the WEM), and where stop-words were included but the similarity threshold was higher for them ($\alpha+0.1$ and $\alpha+0.2$).

Over the total 2800 experiments that were performed the optimal performance was obtained from WEM D (which had a window of 3 and a minimum word count of 10), using a threshold $\alpha = 0.4$ for regular words and $\alpha = 0.5$ for stop-words, and not utilizing corpus weights. The results are displayed in Table 4. That experiment had a correlation on the training data of 0.820, and an MSE of 0.719; both scores were optimal over all experiments. These parameters were therefore chosen for the final measure.

Tables 3 and 4 highlight that, perhaps counter-intuitively, not using corpus weights improved the performance. Across all experiments the correlation was higher and the MSE lower where corpus weights were not used. It may be that the WEM automatically dealt with weighting by treating less-common words like noise, and that therefore when weights were utilized the performance was actually impeded. It was also noted that Model H, rather than Model D would have been chosen as the optimal model were the experiments excluding corpus weights not performed.

The optimal Model, D, had a minimum word count of 10, meaning that it had a smaller number of unique words in the

TABLE 4. Selected results for training excluding corpus weights, with optimal results in bold.

WEM Parameters	Threshold α	Including stop-words		Excluding stop-words		Threshold of α+0.1 for stop-words		Threshold of α+0.2 for stop-words	
		MSE	Cor.	MSE	Cor.	MSE	Cor.	MSE	Cor.
Model B: window = 3, min word = 3	0.1	2.248	0.752	1.983	0.781	2.248	0.752	2.223	0.751
	0.2	2.170	0.730	1.855	0.766	2.119	0.728	2.029	0.746
	0.3	1.398	0.697	1.045	0.760	1.223	0.732	1.096	0.760
	0.4	0.908	0.737	**0.789**	0.780	0.783	0.778	**0.737**	0.796
	0.5	0.953	0.779	1.027	0.792	0.979	0.797	1.027	0.792
	0.6	1.453	0.779	1.504	0.778	1.504	0.778	1.504	0.778
	0.7	1.684	0.771	1.684	0.771	1.684	0.771	1.684	0.771
Model C: window = 3, min word = 5	0.1	2.045	0.750	1.779	0.779	2.040	0.750	2.002	0.747
	0.2	1.903	0.721	1.577	0.752	1.821	0.713	1.723	0.730
	0.3	1.163	0.714	0.876	0.771	1.003	0.746	0.879	0.775
	0.4	0.885	0.744	0.805	0.784	0.747	0.796	0.760	0.796
	0.5	0.957	0.800	1.045	0.802	1.013	0.806	1.045	0.802
	0.6	1.446	0.785	1.478	0.782	1.478	0.782	1.478	0.782
	0.7	1.712	0.768	1.712	0.768	1.712	0.768	1.712	0.768
Model D: window = 3, min word = 10	0.1	1.797	0.759	1.523	0.783	1.775	0.757	1.719	0.758
	0.2	1.520	0.746	1.241	0.762	1.413	0.752	1.346	0.755
	0.3	0.976	0.739	0.812	0.772	0.907	0.750	0.796	0.781
	0.4	**0.843**	0.766	0.794	0.806	**0.719**	**0.820**	0.769	**0.814**
	0.5	1.063	**0.803**	1.140	**0.808**	1.134	0.809	1.140	0.808
	0.6	1.501	0.787	1.506	0.786	1.506	0.786	1.506	0.786
	0.7	1.713	0.768	1.713	0.768	1.713	0.768	1.713	0.768
Model H: window = 5, min word = 10	0.1	1.913	0.744	1.596	0.771	1.896	0.744	1.865	0.743
	0.2	1.658	0.727	1.332	0.750	1.601	0.728	1.483	0.735
	0.3	1.156	0.700	0.915	0.747	1.032	0.724	0.924	0.749
	0.4	0.877	0.747	0.803	0.789	0.767	0.792	0.748	0.804
	0.5	0.990	0.787	1.063	0.796	1.016	0.803	1.063	0.796
	0.6	1.434	0.792	1.488	0.786	1.488	0.786	1.488	0.786
	0.7	1.713	0.768	1.713	0.768	1.713	0.768	1.713	0.768

corpus (429,326) than the other models; it would seem that a smaller corpus, with more unusual words removed (those that occur less than 10 times), together with a small window size (of 3) is therefore preferable for TSSSM.

More generally, it was also noted that the choice of metric is important where determining the optimal model and parameters. Across many experiments a different Model would have been selected were MSE rather than correlation chosen as the metric. This is highlighted in Table 3, where Model B consistently had the optimal results in terms of MSE, but (in all but one scenario) Model H had the optimal results where correlation was considered.

5) ILLUSTRATIVE EXAMPLE

The following example provides a graphical illustration of how the semantic similarity, S_{sem}, is calculated for two fictional tweets. For this example, the optimal parameters as determined in the previous section are utilized. That is using WEM D (which had a window size of 3 and minimum word count of 10), with a similarity threshold of $\alpha = 0.4$ for regular words and 0.5 for stop-words (as defined by Gensim [44]), and no corpus weights. The two tweets are:

- T_1 : "What will the economic consequence of Brexit be?"
- T_2 : "Will #brexit cause the economy problems"

After pre-processing (which in this case would simply convert the words to lower case and remove the question mark), the joint word-set is:

T = {what, will, economic, consequence, brexit, #brexit, cause, economy, problems}.

The words "the", "of" and "be" were part of a small number of common words excluded from the joint word-set.

Table 5 illustrates the process of deriving the semantic vector s_1 for tweet T_1, with all values rounded to 3 decimal places. The words in the joint word-set are in the first column, and the words from tweet T_1 are in the top row. The words common to both have a semantic value of 1 (and the cell at the cross-point is set to a value of 1). The example illustrates how the measure will match identical words that differ only by the presence of a hash symbol; "#brexit" and "brexit" are considered a match and assigned a value of 1.

The words that are not common to both, i.e. those that are only in the joint word-set (and hence unique to tweet T_2) are then compared against all the words in T_1, and the match with the highest similarity is selected. To illustrate this all matches are included in Table 5. For example, the word "cause" was paired with each of the words in T_1 using the WEM and the highest similarity score (0.418) was found to exist between it and "consequence"; as this score was above the similarity threshold ($\alpha = 0.4$) it was included in the semantic vector. In contrast, the word "problems" had no match above 0.4 and therefore a value of 0 was included in the semantic vector.

TABLE 5. Example of the process for deriving the semantic vector for Tweet 1.

Joint word set ↓	Words in tweet T_1								Semantic vector, V_1
	what	will	the	economic	consequence	of	brexit	be	
what	**1**								1
willl		**1**							1
economic				**1**					1
consequence					**1**				1
brexit							**1**		1
#brexit							**1**		1
cause	0.207	0.233	0.187	0.284	**0.418**	0.216	0.221	0.256	0.418
economy	0.243	0.304	0.257	**0.500**	0.208	0.254	0.317	0.286	0.500
problems	0.127	0.163	0.156	0.295	0.298	0.179	0.144	0.166	0

TABLE 6. Example of the process for deriving the semantic vector for Tweet 2.

Joint word set ↓	Words in tweet T_2						Semantic vector, V_2
	will	#brexit	cause	the	economy	problems	
what	0.416	**0.404**	0.207	0.479	0.243	0.127	0.404
willl	**1**						1
economic	0.270	0.230	0.284	0.283	**0.500**	0.295	0.500
consequence	0.238	0.099	**0.418**	0.209	0.208	0.298	0.418
brexit		**1**					1
#brexit		**1**					1
cause			**1**				1
economy					**1**		1
problems						**1**	1

The same process was applied for tweet T_2, and is illustrated in Table 6. In this case, whilst there were two higher semantic matches for the word "what" in the joint word-set, it was paired with "#brexit" as the other matches were stop-words ("will" and "the") and so required to be over the similarity threshold of 0.5.

Two semantic vectors were produced from which the overall semantic similarity could be derived using the cosine coefficient. The vectors are (to 3 decimal places):

$$V_1 = \{1, 1, 1, 1, 1, 1, 0.418, 0.500, 0\}$$
$$V_2 = \{0.404, 1, 0.500, 0.418, 1, 1, 1, 1, 1\}$$

And the resulting semantic similarity score is:

$$S_{sem}(T_1, T_2) = 0.805$$

Indicating fairly high semantic similarity between the two tweets, as might be expected.

B. SYNTACTIC ELEMENT

The syntactic element of TSSSM has two components: it considers the syntactical order of the words; and the presence of hashtags, mentions and pronouns. Whilst the semantic element of the measure can identify common meaning, it does not consider word order or structure, therefore the syntactic element performs this task.

1) LONGEST COMMON SYNTACTICAL SEQUENCE

For each tweet pair the longest common syntactical sequence of words was considered. This identifies pairs of tweets that share a common syntactical structure but will also identify identical sequences of words (for instance, where one tweet might contain a quote from another). The longest common syntactical sequence value is denoted by, S_{lcs}, where:

$$S_{lcs} \in [0, 1] \tag{6}$$

To calculate this Part of Speech (POS) tags were attached to each word in the tweets using the spaCy POS tagger. Each tweet is therefore represented, in sequence, by its POS tags. The spaCy Universal tag set [50] was utilized with some simplification: proper nouns were classed in one overall group of nouns; auxiliary verbs were classed as verbs; and coordinating and subordinating conjunctions were classed as conjunctions. This left twelve possible tags (excluding symbols and punctuation which were removed in pre-processing), these were: Adjectives, Adpositions, Adverbs, Conjunctions, Determiners, Interjections, Nouns, Numerals, Particles, Pronouns, Verbs and Other (anything that does not fit these categories). During the tagging process Twitter usernames/mentions (i.e. @username) were tagged as nouns, and the hash symbol was removed from hashtags to allow the word to be tagged.

Once tweets are POS tagged TSSSM searches the two tweets for common sequences of tags. If there is a sequence greater than two tags (any smaller is deemed not relevant) the length of this (in words/tags) is divided by the length of the shortest tweet (in words/tags). This means that if one tweet's syntactic sequence is entirely contained in the other, the value of S_{lcs} will be 1. If there is no common sequence, then the value of S_{lcs} will be zero.

For example, given tweets

- T_1 : "the man kicked a ball very far"
- T_2 : "the boy threw a frisbee up high"

the POS tags consist of:

- "**DETERMINER NOUN VERB DETERMINER NOUN** ADVERB ADVERB"
- "**DETERMINER NOUN VERB DETERMINER NOUN** ADPOSITION ADVERB"

which share a common sequence of five tags (shown in bold). The length of each of the tweets is seven words, therefore the common syntactic sequence value would be $S_{lcs}(T_1, T_2) = 5/7 = 0.714$, indicating a long commonly shared syntactic sequence.

As a further example, given two tweets that share an identical sequence of words:

- T_1 : "I think that brexit should be given some more thought"
- T_2 : "brexit should be given some more thought"

the POS tags consist of:

- "PRONOUN VERB CONJUGATION **NOUN VERB VERB VERB DETERMINER ADVERB ADJECTIVE**"
- "**NOUN VERB VERB VERB DETERMINER ADVERB ADJECTIVE**"

These share a common sequence of seven tags (shown in bold). As T_2 is contained entirely in T_1 this is reflected in the common syntactical sequence value, which is seven divided by the length of T_2 (the shortest tweet length of the two tweets), that is $S_{lcs}(T_1, T_2) = 7/7 = 1$.

2) FEATURE VECTOR

For each tweet, a syntactic vector is created that counts the number of hashtags, mentions and pronouns, these values are then divided by the number of words in the tweet in order to normalize them. For instance, if a tweet has five hashtags out of a total of ten words, then the hashtag value would be 0.5. In contrast, a tweet having five hashtags out of fifty words would have a value of 0.1; dividing by the total reflects the proportion. Various experiments were performed to determine the features that might be included in the feature vector. For use in a political domain mentions, hashtags and pronouns were deemed useful. Pronouns were considered interesting in that their presence or not can indicate the tone of the tweet (for example, whether it is in the first person, or uses more neutral language).

However, it is envisaged that the feature vector should be flexible – where different features are of interest these may be added in. For instance, there may be some applications where URLs, the presence of media, or other syntactical features may be of interest. These can be easily added to the feature vector.

For each tweet pair, the cosine similarity of the two feature vectors is calculated to determine the level of similarity. This is represented by S_{fv}, where:

$$S_{fv} \in [0, 1] \tag{7}$$

Whilst it was possible to use labelled training data to determine the optimal semantic parameters (WEM, similarity threshold, etc.), it was not possible to do this for the syntactic element, as the training data was not labelled for syntactic similarity. Rather the semantic and syntactic elements were combined, and the training dataset utilized to determine the optimal combination of features.

C. COMBINING THE SEMANTIC AND SYNTACTIC ELEMENTS INTO ONE MEASURE

In order to optimally combine the semantic and syntactic elements various weightings were considered (using the training data to evaluate). Since syntax plays a smaller role in the semantic processing of text [18], it was expected that the semantic element be weighted highest. TREASURE [16] utilized a combination of 0.8 × Semantic + 0.2 x Syntactic. Whereas STASIS [18] used a proportion of 0.85 for the semantic element. Since TSSSM uses three elements (semantic, longest syntactical sequence and syntactic feature vector), it was complex to manually determine the optimal combination of weights, therefore linear regression was utilized on the training data pairs. This identified a formula such that (to 3 decimal places) the overall similarity, *Sim,* is:

$$Sim = 0.045 + (0.910 * S_{sem}) + (0.110 * S_{lcs}) + (0.0339 * S_{fv}) \tag{8}$$

where S_{sem} is the semantic similarity, S_{lcs} is the longest common syntactical sequence value and S_{fv} is the syntactic feature vector similarity value.

This optimal weighting outperformed any other combination, and was utilized in TSSSM, with one caveat: where there was no semantic similarity detected, the overall similarity score was set to zero. It was felt that if zero semantic similarity existed, then this should overrule the syntactic elements. TSSSM produces an overall similarity score, *Sim*, between 0 and 1 (the score was capped at 1, although it occasionally may be higher). For use with the training and testing pairs this was multiplied by 5. TSSSM applies a basic check for identical pairs as the tweets are read in – if a pair are identical, then there is no need to run them through the measure, and their value is set to full similarity.

VI. RESULTS AND EVALUATION

TSSSM was tested on three human-labelled tweet pair sets, as detailed in Section III.B (the Brexit, EU-Referendum and Tweet-News sets). None of the pairs were used in the training of the measure in order to provide an unbiased evaluation.

Table 7 details all 32 results for the Brexit pairs, which were created from the corpus and labelled by 35 human respondents, to test TSSSM. The results are ordered by the mean human similarity score (high to low). TSSSM had a correlation of 0.75 and MSE of 0.52. The high correlation

TABLE 7. Comparing the mean human similarity score to TSSSM score.

Tweet pair	Human similarity (mean)	TSSSM similarity score
Pair 1	4.77	4.98
Pair 2	3.80	3.82
Pair 3	3.67	3.40
Pair 4	3.56	3.35
Pair 5	3.55	2.58
Pair 6	3.54	4.18
Pair 7	3.49	2.39
Pair 8	3.39	3.28
Pair 9	3.29	2.58
Pair 10	3.28	3.56
Pair 11	3.17	2.22
Pair 12	3.06	2.36
Pair 13	3.00	3.45
Pair 14	2.80	2.88
Pair 15	2.79	3.33
Pair 16	2.75	3.37
Pair 17	2.70	2.60
Pair 18	2.44	1.41
Pair 19	2.41	2.59
Pair 20	2.38	3.24
Pair 21	2.26	2.59
Pair 22	2.23	2.86
Pair 23	2.20	3.17
Pair 24	1.83	3.17
Pair 25	1.69	3.03
Pair 26	1.65	2.34
Pair 27	1.49	2.51
Pair 28	1.31	1.53
Pair 29	1.29	1.78
Pair 30	1.05	1.60
Pair 31	0.62	1.64
Pair 32	0.00	0.99

suggests good performance on the unseen test data, and the low MSE indicates that, overall, the predicted similarity scores did not miss the human evaluation by much. This is shown by the fact that for 13 of the pairs, TSSSM's score was within 0.5 of the actual score, and that only 6 pairs had a difference of more than 1 (with the maximum difference between the human and measure similarity score for one pair being 1.3).

The results (MSE and correlation to 2 decimal places) for all three test datasets are listed in Table 8. To provide a comparison the results using the TREASURE [16] measure are also listed. TSSSM outperformed TREASURE in terms of MSE, with much lower scores. TSSSM also had higher correlation for the Brexit and Tweet-News test sets, but an almost identical score for the EU-Referendum pairs. However, TREASURE might be expected to perform well on the EU-Referendum dataset since that set was used to determine its parameters.

Fig. 5 plots the results for the two measures against the mean human scores for each test dataset, with a diagonal line to indicate the ideal score. That is, if the measure and human scores agreed, all dots would be on the diagonal line. The spread of the data indicates that TSSSM generally performed well on the Brexit pairs, and the other two test sets. Although the EU-Referendum test set contained pairs from the same political domain (Brexit) it contained words not in the corpus

TABLE 8. Results and comparison of TSSSM with TREASURE measure.

Dataset	TSSSM		TREASURE	
	MSE	Cor.	MSE	Cor.
Brexit (32 pairs)	0.52	0.75	2.53	0.66
EU-Referendum (30 pairs)	1.1	0.70	3.52	0.71
Tweet-News (688 pairs)	0.75	0.74	1.92	0.55

and therefore there would have been gaps in its semantic relationships; nevertheless, TSSSM performed well.

The test on the Tweet-News pairs provides the biggest test of generalizability. Since the Tweet-News pairs were not from the same domain that the WEM was trained on (containing news, sports, current affairs and popular culture references from 2013) some of the words would not have been contained in the corpus, and the semantic relationships could not have been matched. This is evident in Fig. 5 where a small number of pairs were assigned a score of zero when they should not have been. Yet a correlation of 0.75 represents a good result given that the optimal score for measures trained specifically for the SemEval task for this dataset was 0.79 [20].

In contrast, Fig. 5 indicates that TREASURE generally did not assign lower similarity scores at all; for the Brexit and EU-Referendum pairs the similarities were centered around 3 or 4 and this is reflected in the high MSE.

TSSSM outperformed TREASURE, displaying that it can identify a range of similarities with greater precision. The results on the Tweet-News dataset show that the measure can generalize to examples outside the political domain.

VII. CONCLUSION

The contribution of this paper is a new semantic and syntactic similarity measure, TSSSM, for measuring the similarity of tweets in the political domain. A large dataset of tweets pertaining to Brexit was collected and a word embedding model utilized to learn the semantic relationships between the words in the dataset. The WEM was utilized for the semantic element of the measure and the syntactic element considered sequences of words and features. Considering word order is particularly important as tweets can contain duplication and repeated sequences, and a semantic measure alone may not identify this, potentially leading to tweets being marked as semantically identical when they are not.

TSSSM was evaluated on three human-rated test datasets of pairs and had good results consistent with the human ratings. TSSSM outperformed a similar measure, TREASURE (which did not consider word order), and was also shown to generalize to pairs that were outside of the political domain and containing words not in the WEM corpus.

Returning to the research questions stated at the start of the paper – in consideration of the first question (can a similarity measure be derived for measuring the similarity of political

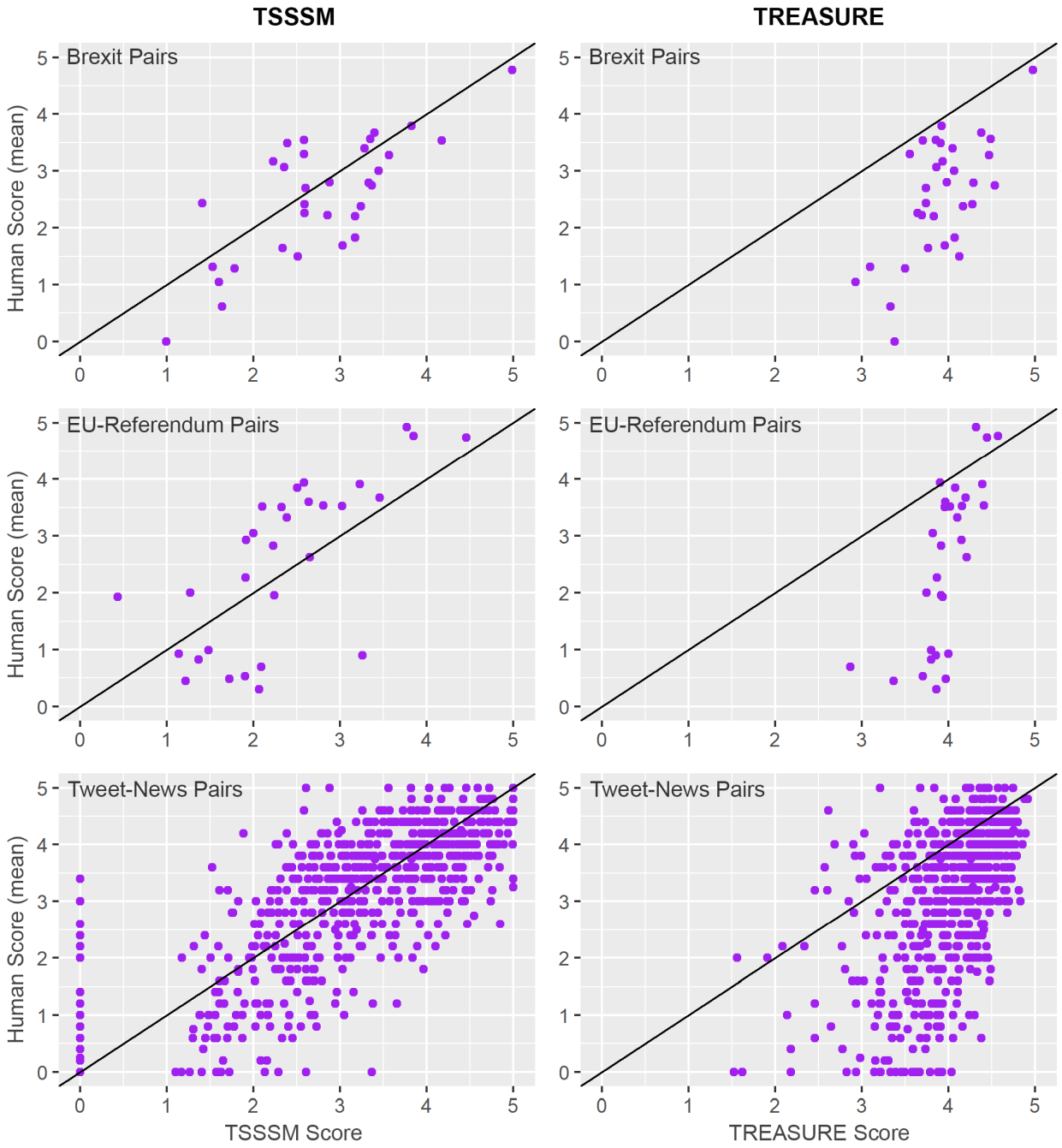

FIGURE 5. Plots comparing the performance of TSSSM with TREASURE on the three test datasets, with a line to indicate perfect correlation.

tweets that embodies both semantic and syntactic information?) the results have shown that TSSSM gives a higher correlation against human ratings than the current similarity measure, TREASURE, on unseen test sets. With regards to the second question (can the developed similarity measure generalize to different domains?) the results across three test sets, in particular for the Tweet-News pairs dataset which was extracted from a different domain to which the WEM was trained on, demonstrate the generalizability of TSSSM.

Further work will include utilizing TSSSM in cluster analysis in order to identify political and populist narratives; and to directly identify similar tweets given examples of tweets typical of these narratives. This would also include identifying narratives containing misinformation and potentially analyzing the types of users who post those kinds of tweets (strictly following ethical practice and Twitter's guidelines). Other work might also explore including a feature within TSSSM to consider the sentiment contained in

tweets, and what effect this has on any narratives identified. Whilst the focus was on the political domain the results show that TSSSM can generalize, however, further work would include improving the measure's performance where it encounters out of vocabulary words. Since WEMs are flexible, new examples can be added to expand the corpus, increase generalizability and experiment with other domains.

ACKNOWLEDGMENT

This work received ethical approval at Manchester Metropolitan University under reference number 8512.

REFERENCES

[1] Twitter, *Q1 2019 Earnings Report Twitter Investor Relations*. Accessed: May 15, 2020. [Online]. Available: https://s22.q4cdn.com/826641620/files/doc_financials/2019/q1/Q1-2019-Slide-Presentation.pdf

[2] (Oct. 2019). *National Politics on Twitter: Small Share of U.S. Adults Produce Majority of Tweets*. Accessed: May 22, 2020. [Online]. Available: https://www.people-press.org/2019/10/23/national-politics-on-twitter-small-share-of-u-s-adults-produce-majority-of-tweets/

[3] K. Jacobs and N. Spierings, "A populist paradise? Examining populists' Twitter adoption and use," *Inf., Commun. Soc.*, vol. 22, no. 12, pp. 1681–1696, Oct. 2019, doi: 10.1080/1369118x.2018.1449883.

[4] U. Klinger and J. Svensson, "The emergence of network media logic in political communication: A theoretical approach," *New Media Soc.*, vol. 17, no. 8, pp. 1241–1257, Sep. 2015.

[5] L. Buccoliero, E. Bellio, G. Crestini, and A. Arkoudas, "Twitter and politics: Evidence from the US presidential elections 2016," *J. Marketing Commun.*, vol. 26, no. 1, pp. 88–114, Jan. 2020, doi: 10.1080/13527266.2018.1504228.

[6] P. L. Francia, "Free media and Twitter in the 2016 presidential election: The unconventional campaign of donald trump," *Social Sci. Comput. Rev.*, vol. 36, no. 4, pp. 440–455, Aug. 2018.

[7] *Digital Technology and the Resurrection of Trust*. Accessed: Jun. 29, 2020. [Online]. Available: https://publications.parliament.uk/pa/ld5801/ldselect/lddemdigi/77/77.pdf

[8] L. Singh, S. Bansal, L. Bode, C. Budak, G. Chi, K. Kawintiranon, C. Padden, R. Vanarsdall, E. Vraga, and Y. Wang, "A first look at COVID-19 information and misinformation sharing on Twitter," 2020, *arXiv:2003.13907*. [Online]. Available: http://arxiv.org/abs/2003.13907

[9] *Covid-19 Disinformation Briefing*. Accessed: May 5, 2020. [Online]. Available: https://www.isdglobal.org/isd-publications/covid-19-disinformation-briefing-no-1/

[10] M. Rooduijn, "The rise of the populist radical right in western europe," *Eur. View*, vol. 14, no. 1, pp. 3–11, Jun. 2015.

[11] S. Engesser, N. Ernst, F. Esser, and F. Bächel, "Populism and social media: How politicians spread a fragmented ideology," *Inf., Commun. Soc.*, vol. 20, no. 8, pp. 1109–1126, Aug. 2017.

[12] *We the People': The Battle to Define Populism*. Accessed: May 23, 2020. [Online]. Available: https://www.theguardian.com/news/2019/jan/10/we-the-people-the-battle-to-define-populism

[13] C. Mudde, "The populist zeitgeist," *Government Opposition*, vol. 39, no. 4, pp. 541–563, 2004.

[14] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.

[15] S. Engesser, N. Fawzi, and A. O. Larsson, "Populist online communication: Introduction to the special issue," *Inf., Commun. Soc.*, vol. 20, no. 9, pp. 1279–1292, Sep. 2017.

[16] N. A. Alnajran, "An integrated semantic-based framework for intelligent similarity measurement and clustering of microblogging posts," Ph.D. dissertation, Fac. Sci. Eng., Manchester Metropolitan Univ., Manchester, U.K., 2019.

[17] M. AL-Smadi, Z. Jaradat, M. AL-Ayyoub, and Y. Jararweh, "Paraphrase identification and semantic text similarity analysis in arabic news tweets using lexical, syntactic, and semantic features," *Inf. Process. Manage.*, vol. 53, no. 3, pp. 640–652, May 2017.

[18] Y. Li, D. McLean, Z. A. Bandar, J. D. O'Shea, and K. Crockett, "Sentence similarity based on semantic nets and corpus statistics," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 8, pp. 1138–1150, Aug. 2006, doi: 10.1109/TKDE.2006.130.

[19] M. Faraon, A. Jaff, L. P. Nepomuceno, and V. Villavicencio, "Fake news and aggregated credibility: Conceptualizing a co-creative medium for evaluation of sources online," *Int. J. Ambient Comput. Intell.*, vol. 11, no. 4, pp. 1–25, Oct. 2020.

[20] E. Agirre, C. Banea, C. Cardie, D. Cer, M. Diab, A. Gonzalez-Agirre, W. Guo, R. Mihalcea, G. Rigau, and J. Wiebe, "SemEval-2014 task 10: Multilingual semantic textual similarity," in *Proc. 8th Int. Workshop Semantic Eval.*, Dublin, Ireland, 2014, pp. 81–91.

[21] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin, "Learning sentiment-specific word embedding for Twitter sentiment classification," in *Proc. 52nd Annu. Meeting Assoc. Comput. Linguistics*, Baltimore, MD, USA 2014, pp. 1555–1565.

[22] Q. Li, S. Shah, R. Fang, A. Nourbakhsh, and X. Liu, "Tweet sentiment analysis by incorporating sentiment-specific word embedding and weighted text features," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell. (WI)*, Omaha, NE, USA, Oct. 2016, pp. 568–571, doi: 10.1109/WI.2016.0097.

[23] N. Al-Twairesh and H. Al-Negheimish, "Surface and deep features ensemble for sentiment analysis of arabic tweets," *IEEE Access*, vol. 7, pp. 84122–84131, 2019, doi: 10.1109/ACCESS.2019.2924314.

[24] M. Baumgarten, M. D. Mulvenna, N. Rooney, and J. Reid, "Keyword-based sentiment mining using Twitter," *Int. J. Ambient Comput. Intell.*, vol. 5, no. 2, pp. 56–69, Apr. 2013.

[25] Y. Madani, M. Erritali, J. Bengourram, and F. Sailhan, "A multilingual fuzzy approach for classifying Twitter data using fuzzy logic and semantic similarity," *Neural Comput. Appl.*, vol. 32, no. 12, pp. 8655–8673, Jun. 2020.

[26] A. Pawar and V. Mago, "Calculating the similarity between words and sentences using a lexical database and corpus statistics," 2018, *arXiv:1802.05667*. [Online]. Available: http://arxiv.org/abs/1802.05667

[27] H. Öztärk and A. Özgär, "BIOSSES: A semantic sentence similarity estimation system for the biomedical domain," *Bioinformatics*, vol. 33, no. 14, pp. i49–i58, Jul. 2017.

[28] T. K. Landauer and S. T. Dumais, "A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge.," *Psychol. Rev.*, vol. 104, no. 2, pp. 211–240, 1997.

[29] T. Magerman, B. Van Looy, and X. Song, "Exploring the feasibility and accuracy of latent semantic analysis based text mining techniques to detect similarity between patent documents and scientific publications," *Scientometrics*, vol. 82, no. 2, pp. 289–306, Feb. 2010.

[30] J. O'shea, Z. Bandar, and K. Crockett, "A new benchmark dataset with production methodology for short text semantic similarity algorithms," *ACM Trans. Speech Lang. Process.*, vol. 10, no. 4, pp. 1–63, Dec. 2013, doi: 10.1145/2537046.

[31] N. Alnajran, K. Crockett, D. McLean, and A. Latham, "An empirical performance evaluation of semantic-based similarity measures in microblogging social media," in *Proc. IEEE/ACM 5th Int. Conf. Big Data Comput. Appl. Technol. (BDCAT)*, Zurich, Switzerland, Dec. 2018, pp. 126–135.

[32] Y. Tian, H. Li, Q. Cai, and S. Zhao, "Measuring the similarity of short texts by word similarity and tree kernels," in *Proc. IEEE Youth Conf. Inf., Comput. Telecommun.*, Beijing, China, Nov. 2010, pp. 363–366.

[33] G. A. Miller, "WordNet: A lexical database for English," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[34] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, Feb. 2003.

[35] Y. Kim, "Convolutional neural networks for sentence classification," 2014, *arXiv:1408.5882*. [Online]. Available: http://arxiv.org/abs/1408.5882

[36] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proc. 2013 Conf. Empirical Methods Nat. Lang. Process.*, Seattle, WA, USA, 2013, pp. 1631–1642.

[37] Q. Li, S. Shah, A. Nourbakhsh, X. Liu, and R. Fang, "Hashtag recommendation based on topic enhanced embedding, tweet entity data and learning to rank," in *Proc. 25th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2016, pp. 2085–2088, doi: 10.1145/2983323.2983915.

[38] F. Godin, B. Vandersmissen, W. De Neve, and R. Van de Walle, "Multimedia lab ACL WNUT NER shared task: Named entity recognition for Twitter microposts using distributed word representations," in *Proc. Workshop Noisy User-generated Text*, Beijing, China, 2015, pp. 146–153.

[39] N. N. Alnajran, K. A. Crockett, D. McLean, and A. Latham, "A word embedding model learned from political tweets," in *Proc. 13th Int. Conf. Comput. Eng. Syst. (ICCES)*, Dec. 2018, pp. 177–183, doi: 10.1109/ICCES.2018.8639217.
[40] X. Yang, C. Macdonald, and I. Ounis, "Using word embeddings in Twitter election classification," *Inf. Retr. J.*, vol. 21, nos. 2–3, pp. 183–207, Jun. 2018.
[41] Tweepy. *Tweepy Documentation*. Accessed: May 16, 2020. [Online]. Available: http://docs.tweepy.org/en/latest/
[42] PyMongo. *PyMongo 3.10.1 Documentation*. Accessed: May 16, 2020. [Online]. Available: https://pymongo.readthedocs.io/en/stable/
[43] F. Morstatter, J. Pfeffer, and H. Liu, "When is it biased? Assessing the representativeness of Twitter's streaming API," in *Proc. 23rd Int. Conf. World Wide Web*, Seoul, South Korea, 2014, pp. 555–556s
[44] P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proc. Workshop New Challenges NLP Frameworks*, Valletta, Malta, 2010, pp. 40–45.
[45] NLTK. *NLTK 3.5 Documentation*. Accessed: May 17, 2020. [Online]. Available: https://www.nltk.org/
[46] W. Guo, H. Li, H. Ji, and M. Diab, "Linking tweets to news: A framework to enrich short text data in social media," in *Proc. ACL*, Sofia, Bulgaria, 2013, pp. 239–249.
[47] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, *arXiv:1301.3781*. [Online]. Available: http://arxiv.org/abs/1301.3781
[48] M. Bansal, K. Gimpel, and K. Livescu, "Tailoring continuous word representations for dependency parsing," in *Proc. 52nd Annu. Meeting Assoc. Comput. Linguistics*, Baltimore, MD, USA, 2014, pp. 809–815.
[49] *spaCy*. Accessed: May 26, 2020. [Online]. Available: https://spacy.io/
[50] spaCy. *Part-of-Speech Tagging*. Accessed: Jun. 6, 2020. [Online]. Available: https://spacy.io/api/annotation#pos-tagging

CLAIRE LITTLE (Member, IEEE) received the B.Sc. degree (Hons.) in mathematics from The University of Manchester, U.K., in 2002, the M.Sc. degree in computing from Manchester Metropolitan University, Manchester, U.K., in 2012, and the Ph.D. degree in machine learning, in 2018.

She is currently a Research Associate with the Centre for Policy Modelling, Manchester Metropolitan University. Her research interests include machine learning in particular, applying these techniques to social science data, artificial intelligence, data analysis and visualization, and conversational agents.

DAVID MCLEAN received the B.Sc. degree (Hons.) in computer science from the University of Leeds, in 1989, and the Ph.D. degree in neural networks (generalization in continuous data domains) from Manchester Metropolitan University, in 1996.

From 1996 to 1997, he was with DERA, Malvern, and Thomson Marconi Sonar. He became a Lecturer with Manchester Metropolitan University in 1997. He currently leads the M.Sc. Digital Technology Apprenticeship Program and Lectures on programming, data structures, and data mining units. He holds a patent (Analysis of the Behaviour of a Subject). He has previous involvement with spin-out companies involved with psychological profiling and conversational agents. His research interests include text mining and short text similarity measures in particular.

KEELEY CROCKETT (Senior Member, IEEE) has over 20 years' experience of research and development in computational intelligence algorithms and applications, including adaptive psychological profiling, fuzzy systems, dialogue systems, and educational tutoring systems. She is currently a Reader in computational intelligence with Manchester Metropolitan University and a Leader with the Intelligent Systems Group. She is also a member of the Taskforce on Ethical and Social Implications of Computational Intelligence. She is a co-investigator on the H2020 Project PACE. She has coauthored over 120 peer-reviewed publications. She serves as the Chair for the IEEE Women in Computational Intelligence and is a U.K. STEM Ambassador.

BRUCE EDMONDS received the B.A. degree (Hons.) in mathematics from the University of Oxford, in 1983, and the Ph.D. degree in philosophy of science (syntactic measures of complexity) from The University of Manchester, in 1999.

With his colleague S. Moss, he developed the Centre for Policy Modelling (CfPM) into one of the world leading in agent-based social simulation. He is currently a Professor of social simulation and the Director of CfPM, Manchester Metropolitan University. He is the Coordinator of the H2020-Funded Populism and Civic Engagement (PaCE) Project and the PI of the U.K. part of the Towards Realistic Computational Models of Social Influence Dynamics (ToRealSim) Project. He co-edits the handbook on *Simulating Social Complexity* (Springer, Second Edition) with his colleague R. Meyer.